Constraints First:
A New MDD-based Model to Generate Sentences Under Constraints

Alexandre Bonlarron\textsuperscript{1,3}, Aurélie Calabrèse\textsuperscript{2}, Pierre Kornprobst\textsuperscript{1} and Jean-Charles Régis\textsuperscript{3}
\textsuperscript{1}Université Côte d’Azur, Inria, France
\textsuperscript{2}Université Aix-Marseille, CNRS, LPC, France
\textsuperscript{3}Université Côte d’Azur, CNRS, I3S, France
alexandre.bonlarron@inria.fr, aurelie.calabrese@univ-amu.fr, pierre.kornprobst@inria.fr, jean-charles.regin@univ-cotedazur.fr

Abstract
This paper introduces a new approach to generating strongly constrained texts. We consider standardized sentence generation for the typical application of vision screening. To solve this problem, we formalize it as a discrete combinatorial optimization problem and utilize multivalued decision diagrams (MDD), a well-known data structure to deal with constraints. In our context, one key strength of MDD is to compute an exhaustive set of solutions without performing any search. Once the sentences are obtained, we apply a language model (GPT-2) to keep the best ones. We detail this for English and also for French where the agreement and conjugation rules are known to be more complex. Finally, with the help of GPT-2, we get hundreds of bona-fide candidate sentences. When compared with the few dozen sentences usually available in the well-known vision screening test (MNREAD), this brings a major breakthrough in the field of standardized sentence generation. Also, as it can be easily adapted for other languages, it has the potential to make the MNREAD test even more valuable and usable. More generally, this paper highlights MDD as a convincing alternative for constrained text generation, especially when the constraints are hard to satisfy, but also for many other prospects.

1 Introduction
We consider the problem of generating text under constraints. Currently, the methodology for solving this problem is based on two main components: (a) generating words using Language Models (LM) that have been proven on several tasks in natural language processing [Wang et al., 2018]\textsuperscript{1}; (b) processing text sequentially to satisfy constraints which can be done by tokens (words) filtering [Roush et al., 2022] or search mechanisms such as Beam Search (BS) [Liu et al., 2021; Post and Vilar, 2018; Hokamp and Liu, 2017] or more recently A* [Lu et al., 2022]. These search-based heuristics involve generating the text word-by-word while maximizing the likelihood of the sequence computed by (a), exploring solutions space, and checking constraints with (b). This methodology is appropriate as long as the solution space is weakly constrained, i.e., when it is easy to find a sequence satisfying the constraints. This is usually the case for lexical or semantic constraints since they are local requirements in the output and are almost formalized as output preferences.

In this paper, we are interested in generating constrained text when the solutions space is strongly constrained (i.e., when it is hard to find a solution that satisfies the constraints). This is the case when constraints are not local anymore (e.g., length constraint, display constraint). In other words, they are defined on the whole text. Because the sequences we are looking for are rare, we propose a completely different methodology based on three ideas: (i) We propose to check the constraints first and then select the best solutions with an LM because there is no reason that the sequences that an LM considers likely would satisfy the constraint first. (ii) We see a constrained text generation problem as a discrete combinatorial optimization problem, where variables are words, the domain of variables is the vocabulary of text, and constraints are the set of rules the text has to satisfy. (iii) Even though we want to satisfy the constraints first, applying a pure generate and test over random words would find sequences satisfying the constraints, but those sequences would not have a meaning. In other words, these sequences are not "well-formed" sentences. Since we cannot totally ignore the language structure, we rely on combining existing sequences of words (n-grams). For example, from the sentence: "He lives in a nice little red house", we can find several 3-grams (e.g., "a nice little", "nice little red", "little red house"). On top of that, we can link two n-grams if the \( n-1 \) last words of the first n-gram are the \( n-1 \) first words of the second n-gram. Then sentences are obtained by following links between n-grams. Note that this idea of combining n-grams was used to generate palindromes (e.g., race car) [Papadopoulos et al., 2015] with an ad-hoc algorithm.

Our contribution is twofold: we propose a Constraint Programming (CP) model for sentence generation and apply our model to generate standardized sentences for the MNREAD test [Mansfield et al., 1993], a well-known test in vision research for measuring reading performance. MNREAD sentences must satisfy a sharp protocol (i.e., a set of strict rules), and they are hard to find. To be convinced, out of a sample of more than three million sentences from children’s literature

\textsuperscript{1}https://gluebenchmark.com/
(600 books), we only found four obeying all rules.

Our approach is based on multi-valued decision diagrams (MDDs) to produce sentences. An MDD is a well-suited data structure to store and retrieve successions of n-grams. Also, using MDD provides a way to compute an exhaustive set of solutions without performing a search. Therefore, our approach is an exact method. In addition, we use a Language Model (LM) to select the best sentences. An LM is a probabilistic model used to predict the sequence of words in a text or speech using natural language training data. Here we use transformers-based LM (i.e., LM implementing Transformers architecture based on attention block [Vaswani et al., 2017]). More specifically, we use GPT-2 [Brown et al., 2020], which has been trained over a tremendous amount of text and is designed to generate sequences. GPT-2 is also suited to assess sentence quality to a certain degree by assigning a score to sequences. We use this score to sort sentences and select the best ones. It drastically simplifies human verification and, in the best case, avoids it.

The paper is organized as follows: we give some preliminaries in Sec. 2, with a presentation of the MNREAD rules and MDDs. Then in Sec. 3, we show how to model the constrained text generation problem with MDDs. We also explain how to use an LM to select the best sentences. The experimental results are presented in Sec. 4, in which we show the potential of our method. At last, in Sec. 5, we share additional thoughts on this work and give some perspectives on future investigations. Finally, we conclude.

2 Preliminaries
2.1 Case Study: MNREAD Test

Motivation
Reading performance has become one of the most widely used clinical measures to judge the effectiveness of treatments, surgical procedures, or rehabilitation techniques. This reading performance is measured by the time taken to read standardized texts, i.e., texts designed to be equivalent in terms of length, display, and linguistics. This need for equivalent texts is the key to avoid bias in the texts themselves. In addition, each text should only be presented once to the subjects or patients to avoid recall bias and to ensure an accurate measurement.

Among various existing tests, the MNREAD test [Mansfield et al., 1993] is probably one of the most widely used standardized reading tests in the world for measuring reading performance in clinical and research settings in people with normal or low vision. For instance, it is used to evaluate how reading performance depends on font size [Calabrese et al., 2016]. One MNREAD test comprises 19 standardized sentences printed in decreasing sizes. It is available in 19 languages, with different numbers of tests depending on the language (e.g., five in English, two in French). Since repeated measurements are needed, a large number of test sets are required. Unfortunately, currently, the number of available MNREAD sentences is largely insufficient.

Related Work
Some researchers designed ad-hoc approaches dedicated to reading tests [Crossland et al., 2008; Perrin et al., 2015; Mansfield et al., 2004]. In particular, Mansfield et al. [Mansfield et al., 2019], who are also the creators of the MNREAD test, proposed an approach based on sentence templates capable of generating millions of sentences using a random walk. However, this semi-automatic method has four major drawbacks: (i) It relies on sentence templates that have to be created manually (i.e., sequences of placeholders, each containing a list of possible words that fit into the sentence at a defined position). (ii) It requires a manual verification of the sentences generated by the system. (iii) It cannot be easily extended to other languages, such as French, where one has to take into account agreements and conjugations or other languages that use declension (e.g., German, Slavic, ...). (iv) Since the sentences generated are based on templates, the number of different words is limited. It may lead to a memory bias for readers.

Characterisation of MNREAD Sentences: Definition of Rules
In order to be standardized, all MNREAD sentences must satisfy five rules $[R_{0}]..[R_{4}]$. The sentence is displayed with left and right justification on three lines of text and must fit in a box whose width is 17.3 times the size of a standard Times-Roman letter. The width of the spaces between the words (shown as a multiple of the normal width of the spaces) should be between 0.80 and 1.25.

An example of an MNREAD sentence respecting these rules is given in Fig. 1 (adapted from [Mansfield et al., 2019]).

Figure 1: An MNREAD sentence that satisfies the constraints $[R_{0}]..[R_{4}]$. The sentence is displayed with left and right justification on three lines of text and must fit in a box whose width is 17.3 times the size of a standard Times-Roman letter. The width of the spaces between the words (shown as a multiple of the normal width of the spaces) should be between 0.80 and 1.25.
MNREAD if it complies with all the rules previously stated, makes sense and finally, resembles the official sentences.

2.2 Multi-Valued Decision Diagram

MDDs are a generalization of binary decision diagrams (BDDs) [Akers, 1978]. MDDs have already been successfully used in many other problems, such as the generation of music [Roy et al., 2016] or poems [Perez and Régis, 2017]. MDDs are usually a background data structure in CP solver initially introduced for table constraints [Cheng and Yap, 2010; Lecoutre, 2011] and are sufficiently generic to compute and store almost any constraint [Wang and Yap, 2022; Gentzel et al., 2022; Verhaeghe et al., 2019; Verhaeghe et al., 2018]. Other uses of MDDs for optimization problems can be found in [Bergman et al., 2016; Perez, 2017; Gillard and Schaus, 2022; Rudich et al., 2022; van Hoeve, 2022].

They are data structures for computing and storing solutions (tuples) of a problem using a directed acyclic graph (DAG). MDDs allow representing a large number of solutions in a compressed structure.

Using MDDs in our context offers three main advantages: (i) MDD is a straightforward extension and an efficient way of compactly representing a corpus (i.e., the succession of words/n-grams of a text). (ii) MDD is a very powerful modeling tool. (iii) It allows the computation of an exhaustive set of solutions without performing a search.

In this paper, we consider deterministic reduced ordered MDDs [Amilhastre et al., 2014]. MDDs are structured in layers where each layer represents an ordered variable $X_i$. There are two particular nodes in the MDD: the root (root) and the true terminal node (tt). Each path between the node root and tt forms a label tuple corresponding to an assignment of the variables associated with each layer. More precisely, the label $w_{a_k}$ of the arc $a_k$ for the variable $X_i$ means that the variable $X_i$ is assigned to $w_{a_k}$, and a path $p = (a_1, a_2, a_3, \ldots, a_r)$ corresponds to the global assignment $(w_{a_1}, w_{a_2}, w_{a_3}, \ldots, w_{a_r})$ of the variables (i.e. an $r$-uplet, or a in our context a sentence of $r$ words).

3 Our Approach

First, we define our input data (i.e., an n-grams set). Then, we explain how to efficiently manipulate these n-grams using MDDtrie, a specific MDD. Next, we describe how to compile the final MDDMNREAD thanks to the MDDtrie to obtain the solution sets (i.e., the generated sentences). Finally, we explain how we use an LM to sort sentences.

3.1 Definition of the Input

Let us consider a set of n-grams, extracted from sentences gathered in a corpus (a set of books). The specificity of our application is that we will ignore a certain number of sentences (and therefore of n-grams) in order to integrate the $R_0$ grammatical rules from the beginning (e.g., interrogative sentences will be ignored, in order not to add n-grams corresponding to interrogative turns of phrases which are not accepted as MNREAD sentences). Finally, thanks to the part-of-speech tagging feature of treetagger library [Schmid, 2013], we filter out any n-gram that does not fit the lexical rule ($R_1$) and also punctuations (e.g., ellipsis, colon, and semicolon and inverted commas). In addition, we differentiate between three types of n-grams: initial n-grams (starting a sentence), middle n-grams, and final n-grams (ending a sentence). In this way, depending on the position of the word in the sentence we are trying to assign, some n-grams are allowed and others are not. We explain in the following sections how to recombine them while satisfying constraints.

3.2 Defining MDDs Associated With Constraints

MDDtrie: the MDD of the Succession Constraint

The goal of MDDtrie is to define a succession rule between words. To store and retrieve our n-grams and their successors efficiently, MDDtrie contains all n-grams of a fixed length. In order to build it, we insert all n-grams as paths (i.e., solutions) and apply the reduction operator (i.e., preduce, [Perez and Régis, 2015]). As a result of MDD properties, each n-gram that shares the same prefix of words shares the same prefix path from root to tt.

Given that we can link two n-grams if the $n - 1$ last words of the first n-gram are the $n - 1$ first words of the second n-gram, finding all words successors of an n-gram $w_1 w_2 w_3 \ldots w_n$ is equivalent to searching in MDDtrie the list of labels of outgoing arcs of the node that can be reached from the root with the subpath $(w_2, w_3, \ldots, w_n)$ (i.e., the suffix of size $n - 1$).

MDDtrie is a different way of defining a gliding constraint with an MDD (see for instance [Jung and Régis, 2022] in the case of the sequence constraint).

An example of the use of MDDtrie is illustrated in a simplified case in Fig. 2.

![Figure 2: Example of MDDtrie storing 3-grams (successions of 3 words): "The black cat"; "A red apple"... Any path from the root to tt is a valid n-gram. To find the successors of the n-gram "The white cat", more precisely the following potential words, we start from root a walk along the arcs that contains the labels of the two last arc, i.e., "white" and "cat". In that case, one outgoing arc from the node can be reached with "white cat". Thus, the successor of "The white cat" is "loves" (in green).](image-url)
MDDMNREAD: The MDD of the Rules $\mathcal{R}_2 \cap \mathcal{R}_3 \cap \mathcal{R}_4$

This new MDD is an unfolded MDD of the MDDTrie of size 15 ($\mathcal{R}_2$). While unfolding it (i.e., applying the succession rule), we solve the constraint problem induced by MNREAD sentence rules (see Fig. 3).

It will contain all possible combinations of words and indirectly of n-grams leading to the production of sentences. The n-grams of the first (resp. last) layer must correspond to the n-grams of the start (resp. end) of the sentence. This eliminates sentences that do not end correctly by avoiding ending on a word that is supposed to introduce other words. As a result, some words are defined as non-terminal (e.g., and, or, to...). The empty word is only possible if the node is an empty word following it. This trick makes it possible to build an MDD containing sentences of different sizes.

In the following paragraphs, while expanding MDDMNREAD, we describe how to compute the remaining constraints. Keep in mind that the label of arcs of the MDDMNREAD is always a word. We also associate costs with each arc. A cost is computed on the fly in relation to the considered constraint (e.g., length, size of the word). So MDDMNREAD is also virtually a cost-MDD.

For constraint $\mathcal{R}_3$: we ensure that each sentence has 59 characters, including spaces by imposing a sum constraint [Trick, 2003] on the length of sentences. In this case, the cost of an arc is the number of characters in a word. Then, MDDMNREAD could be seen as a particular sum MDD whose sum of the labels of the arcs that constitute any solution is equal to 59. (see in Fig. 3 where cost is associated with the label, and the partial sum is propagated through nodes).

For constraint $\mathcal{R}_4$: We ensure the display rule (see example in Fig. 1), which states that a sentence must be displayed on three lines, by imposing a sum constraint on the width of the characters of the sentence. The width of a character is defined according to the font used. In this case, the cost of an arc is the width of a word in its font. Because we want three lines, we have to check three different sums (one for each line). To do so, each time the sum associated with a line is valid, the sum is reset to 0.

Once computed, MDDMNREAD contains sentences that satisfy all rules ($\mathcal{R}_i$), $i=0,..,4$.

### 3.3 Sentences Selection by LM

Unfortunately, n-grams only partially handle the meaning and grammatical concerns of the sentences. Indeed they have limitations, especially in the case of long-range dependencies (i.e., where the dependency size between two words exceed the n-grams size). Therefore, we propose to use an LM to select candidate sentences (correct sentences with meaning) inside the solutions space.

**Model Choice: a Generative Model**

There are a lot of transformers-based LM due mainly to their effectiveness. These large LMs are trained on several tens of gigabytes of text. These texts are cut into chunks of thousands tokens, and LMs process these sequences of tokens to learn a text distribution. Once trained, generative models (like GPT-2) can generate sequences using a part of existing sequences or a token-of-start to compute a distribution of potential successors. Thus, one of the main goals of an LM is the capability to compute the probability of a sequence based on training data. We choose to use GPT-2 because it is recognized as one of the best in language modeling tasks on several benchmarks².

**Sentences Scoring: Perplexity**

As a matter of fact, from GPT-2, which computes the probability of a sequence, we can compute a perplexity score of that sequence. Perplexity is an entropy metric derived from Shannon’s information theory [Brown et al., 1992]. It can be expressed as the geometric mean of the inverse conditional likelihood of the sequence [Jurafsky and Martin, 2009]. Given $S_n$ the sequence of a succession of words of size $n$, so $S_n = w_1w_2...w_n$. The perplexity (PPL) of $S_n$ is computed as follows:

$$PPL(S_n) = \sqrt[n]{\frac{1}{P(w_1w_2w_3...w_n)}}$$

where probability $P(\cdot)$ is given by the LM. PPL can be interpreted as the “uncertainty” of the model with respect to a sample. Usually, it is used to evaluate the LM itself by checking that good samples are recognized as such (i.e., low PPL values).

²https://paperswithcode.com/paper/language-models-are-supervised-multitask#results
GPT-2 as an Oracle
We suppose that GPT-2 has grasped a part of the general language and that its score is trustworthy. Thus, we compute the perplexity of the sentences as a score. This score will assess the correctness and fluency of the sentences. So, if GPT-2 assigns a low PPL to a sentence, we conclude that it is very likely to be a “good” sentence. By “good” sentence, we mean a grammatically correct sentence that makes sense. Once the sentences are generated, we import a pre-trained model of the language of the sentences generated and we sort them according to the PPL score.

4 Results

4.1 Experimental Conditions
The model described in Sec. 3 was implemented in Java 17. The code is available upon request. The experiments were performed on a machine using an Intel(R) Xeon(R) W-2175 CPU @ 2.50GHz with 256 GB of RAM and running under Ubuntu 18.04.
The sentence selection task was performed with models without any fine-tuning by using either OpenAI GPT-2 [Brown et al., 2020] for English sentences or a French trained GPT-2 [Simoulin and Crabbé, 2021]. These models are available from the huggingface library [Wolf et al., 2020].

Target Language
Our primary concern is the French language. It is a hard Latin language. We also generate sentences in English as a proof-of-concept of the multilingual aspect of our approach.

Corpus of N-grams
The implications of corpus construction are developed in Sec. 5.4.
For French, to build our corpus of n-grams, we started with 443 books belonging to the youth category. This choice was motivated by generating sentences having a simple structure and using a simple lexicon (see $R_1$).
For English, to constitute our corpus, we build a set of 75 books fiction category. Nevertheless, we did not “fine-tune” our corpus as much as for French since our goal was obviously to show the multilingual potential of our approach than to produce the best sentences in English.

4.2 Performance Analysis
Table 1 summarizes MDDs computational data. Our model is fast. Some effort was made in this direction for responsiveness. MDDTrie requires a non-negligible amount of memory to be computed (30GB). However, it is deeply related to the fact that strings are generally memory intensive and especially in Java. Nevertheless, since MDDTrie is almost a form of preprocessing of the data input, we do not think that is a critical part for the moment.

<table>
<thead>
<tr>
<th>MDD</th>
<th>arcs</th>
<th>nodes</th>
<th>solutions</th>
<th>GB</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDDT</td>
<td>4,972,698</td>
<td>1,171,904</td>
<td>3,981,618</td>
<td>30</td>
<td>66</td>
</tr>
<tr>
<td>MDDM</td>
<td>23,943</td>
<td>18,983</td>
<td>7,028</td>
<td>72</td>
<td>3</td>
</tr>
<tr>
<td>MDDT</td>
<td>981,372</td>
<td>268,291</td>
<td>735,928</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>MDDM</td>
<td>1390</td>
<td>1201</td>
<td>204</td>
<td>$\leq$ 1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: Number of arcs, nodes, solutions, gigabytes (GB), and seconds (s) for computing MDDTrie (MMDT) and MDDMNREAD (MDDM) from 5-grams. Lines 2 and 3 are for French and 4 and 5 for English.

4.3 Sentence Analysis
Tables 2 and 3 show a sample of some sorted sentences computed from the solution set generated by MDDMNREAD. This sample includes the lowest and largest PPL values. Interestingly, PPL scoring succeeds in sorting sentences. Excellent sentences can be found in the first positions, and very bad sentences can be found in the last positions.
Identifying multiple threshold values, each of which delineates a range within the set of solutions is not straightforward. We observe that we have perfect sentences between the lowest PPL values and PPL less than 15. Between 15 and 30, it gets complicated with good sentences and sometimes few strange sentences. For PPL values greater than 30, we still find some excellent sentences, but we need to become cautious, so it is as if we were selecting sentences without help.
Since the PPL range is $[1;+\infty]$ it is harder to give a confidence interval for the last position because the span between the two last values can be greater than the span between the first sentence and the before-last sentence valuation.
Also, PPL score is strongly dependent on the vocabulary and a fortiori the LM (i.e., training data). Therefore with another LM, we have no guarantee that the PPL min/max and, more generally, the valuation would be similar.
Finally, PPL valuation through GPT-2 is slow. French sentences scoring takes roughly 1 hour.
The system tends to generate all the variants with the same sentence start (see Tab. 4). These variants can all be of good quality, as seen with the beginning: “Le policier passa la main dans sa poche”. However, since we consider the PPL score, only two first sentences would be considered for MNREAD use. PPL helps to choose the best sentences between variants of the sentences.
An established observation in NLP is that lower-size n-grams used to construct text tend to result in a lower grammatical and semantic quality of the output. In this paper, we choose to present benchmarks based on 5-grams. It is possible to decrease the n-gram size further to 4-grams, although this would substantially increase combinatorics. Furthermore, we observe that sometimes good sentences receive poor scores and vice versa. Reducing the n-gram size would result in a way higher occurrence of this phenomenon.
In conclusion, using higher n-gram sizes (4 or 5) to construct sentences is advisable. However, as LMs performances continuously improve, it may become possible to decrease the n-gram size according to the requirements of specific tasks. Nevertheless, be careful when considering lower-size n-grams due to their skyrocketing combinatorics.
4.4 Sentence Evaluation

The purpose of MNREAD sentences is to be read at the same speed. We want to make sure that the generated sentences are equivalent to the original ones. This sentence evaluation allows us to validate our approach, particularly the use of the PPL.

Experimental Design

We asked 14 normally sighted participants (age 14 to 56) to read 3 sets of 30 French sentences: MNREAD sentences, generated sentences ranked as ‘good’ (denoted by low_PPL) and generated sentences ranked as ‘bad’ (denoted by high_PPL). All sentences were displayed at 40cm, in regular polarity, with a fixed print size. Corrected reading speed was measured in words/min (wpm) and analyzed using a mixed-effects model.

Statistical Analysis

The results are shown in the Fig. 4. On average, ‘low_PPL’ generated sentences were read at 164 wpm. This value was not significantly different from the reading speed of 162 wpm yield by MNREAD sentences ($p = 0.5$). On the other hand, reading speed was significantly slower for ‘high_PPL’ generated sentences, with an average value of 151 wpm ($p < 0.001$). Therefore, this preliminary study suggests that our method provides valid standardized sentences that follow the MNREAD rules and yield similar performance, at least in French.

PPL helps in two ways: (i) First, we were able to select good sentences for the MNREAD application. (ii) As a side effect, PPL may be a good predictor of reading speed. Therefore, high PPL will tend us to think that the sentences are not suited for the test because their reading speed may be lower than canonical ones.

5 Discussion

5.1 Post-generation Selection

Since we use n-grams in our approach, our first idea was to build the distribution of the n-gram language model [Jurafsky and Martin, 2009] from the empirical occurrences of words in the input corpus (i.e., perform Markov sampling on the solution set). Many works have been done in this spirit, especially in music generation [Pachet and Roy, 2011]. It gave better results than a random selection but lacked robustness. The drawback of Markov sampling is that it evaluates the meaning of a sentence only from the constrained solution set, whereas it should consider the whole language instead.

We choose to evaluate solutions after the generation stage for two reasons: (i) The transformer’s valuation is time-consuming due to the quadratic complexity architecture. Even though some work is currently done to build a new architecture to reduce its algorithmic complexity (e.g., forecasting time series [Zhou et al., 2021]). (ii) The perplexity is the geometric mean of the inverse conditional probability of the sequences. It means that we need the entire sentence to obtain the exact valuation. This also means that if we want to score all subsequences (subsentences), it will produce a tremendous
amount of requests to the LM at the generation stage because millions of them are produced. Nevertheless, we used it as a powerful oracle for scoring generated sentences.

5.2 CP-ML Hybridization: Perplexity as a Constraint

Currently, the perplexity computed by an LM is not integrated into the model as an additional constraint. This is not an issue because the number of generated sentences is in the order of a few hundreds or thousands and, therefore, not too large. However, millions of sentences could be generated if the constraints were relaxed or if the input corpus size were significantly increased. Evaluating millions of sentences, even on a powerful machine with GPT-2, can take several weeks.

To define a constraint of perplexity $PPL(X < K)$, it is necessary to be able to decide that for a partial assignment of variables (e.g., the variables from 1 to $k$), one can ensure that the total perplexity (for the variables from 1 to $n$) will necessarily exceed $K$. Thus, in this case, we will reject the partial assignment by not continuing it. The major problem is that the perplexity involves all the variables, and the relation between the perplexity of a subset of variables and the full set of variables is not clear. It is not monotonic and can vary greatly and, in particular, improve significantly. Therefore, designing a perplexity constraint is not straightforward.

5.3 Genericity of the Method

A more ad-hoc approach could have been used. However, thanks to MDD, particularly the second one, our method can be easily generalized to integrate other constraints or deal with similar problems. (i) Additional constraints can be integrated similarly to those we have considered (e.g., all-diff, gcc), i.e., during the construction of the final MDD from MDDrie. (ii) We can be even more generic by proceeding by intersections of MDD. Each constraint is associated with an MDD. Then they are integrated with the unfolded MDDrie by intersecting the MDDs. Our first model was originally implemented in this way. This alternative is highly modular and generic because we can obtain, store and reuse any intermediate results. However, it is less efficient in time and memory (counterintuitively, the intersection of two MDDs can take more memory than the two original MDDs because of the local decompressions that can arise). For instance, we needed 64 GB and 6 hours to run the experiments instead of 32 GB and 2 minutes with the current approach.

5.4 The Choice of the Corpus

The choice of the corpus is not critical but deserves particular attention because it strongly impacts the kind of generated sentences.

For example, we tested Wikipedia as an input corpus. The sentences’ tone appears to be “cold”, very impersonal and descriptive (like “This was one of the most famous in the history of the sport” or “It may also be used in the same way as the rest of the army” or “Finalmente le cœur de la ville est située au nord et au sud”). This comes from the fact that our method is based on n-grams and n-grams tend to produce text in the style of an author [Papadopoulos et al., 2014] and because Wikipedia is an encyclopedia. These sentences have not been considered well suited for a vision test.

Another issue was found when we tested a subpart of Google n-grams corpus [Michel et al., 2011]. The Google n-grams represent a large part of the text written in several languages with a wide variety of sources (Google books database). This creates problems because some sentences will lead to the production of n-grams which are not acceptable for our use (e.g., bad n-grams of end of sentences). For instance: “You... you’re not allowed... Dad told you that you shouldn’t do ma... magic...”. Ad-hoc and non-obvious filters must be defined and applied to a considerable amount of data (terabytes). The task is much more complex than it seems.

For all of these reasons, we chose to remain on books for the moment. However, books must also be chosen carefully, because word occurrences are essential. For instance, consider the word “tomato”. Suppose there are not enough books that tell stories about “tomato” in the input corpus. Then the model cannot produce sentences containing the word “tomato”. This observation was made when we put some books about french history. As soon as we did it, we generated sentences with the words “king”, “queen”, “castle”... When creating the corpus, we should therefore consider all the books and the possible relations between them. To go even further in the reasoning: Is it reasonable to mix n-grams from two very far topics like science fiction and heroic fantasy books, allowing the emergence of sentences mentioning “dragon knight flying in the depths of the universe”?

To conclude this section, the choice of the corpus seems to be more an artistic choice than an methodological one.

6 Conclusion

A novel and creative approach for constrained text generation has been presented. It formalizes the problem as a discrete combinatorial optimization problem and solves it using a model based on MDDs. We introduce MNREAD sentence problem generation as a constrained text generation task and we successfully applied our method, allowing the creation of new sets of sentences for MNREAD test. We have generated thousands of sentences in French and hundreds of sentences in English in roughly two minutes. These results will lead to broader use of the MNREAD test. In addition, we show that GPT-2 can be used to cleverly sort the generated sentences to simplify human verification of generated sentences. Our general method can generate any standardized text content, especially with hard constraints. It also brings an outside-the-box point of view on the constrained text generation task by solving the constraints first.

Acknowledgments

This work has been supported by the French government, through the 3IA Côte d’Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P31A-0002.
References


